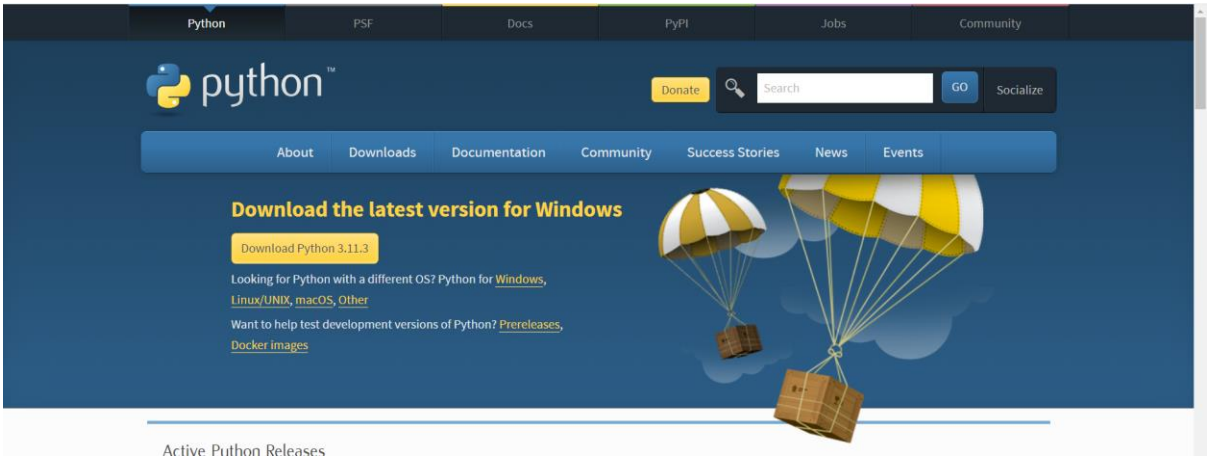
Practical Number: 1

a) Aim: Install NLTK.

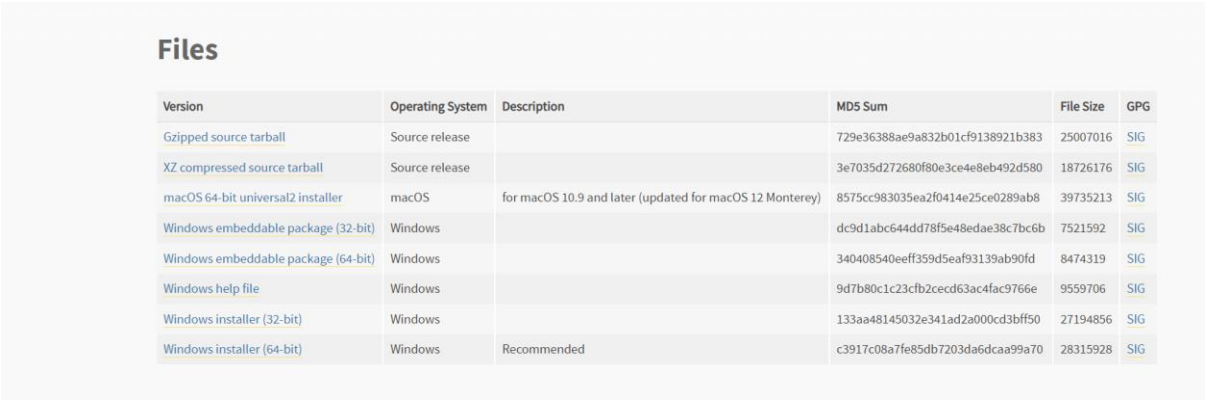
Python 3.11.3 Installation on Windows

Steps:

Step 1) **Go to link** https://www.python.org/downloads/, **and select the latest version for windows.**



Note: If you don't want to download the latest version, you can visit the download tab and see all releases.



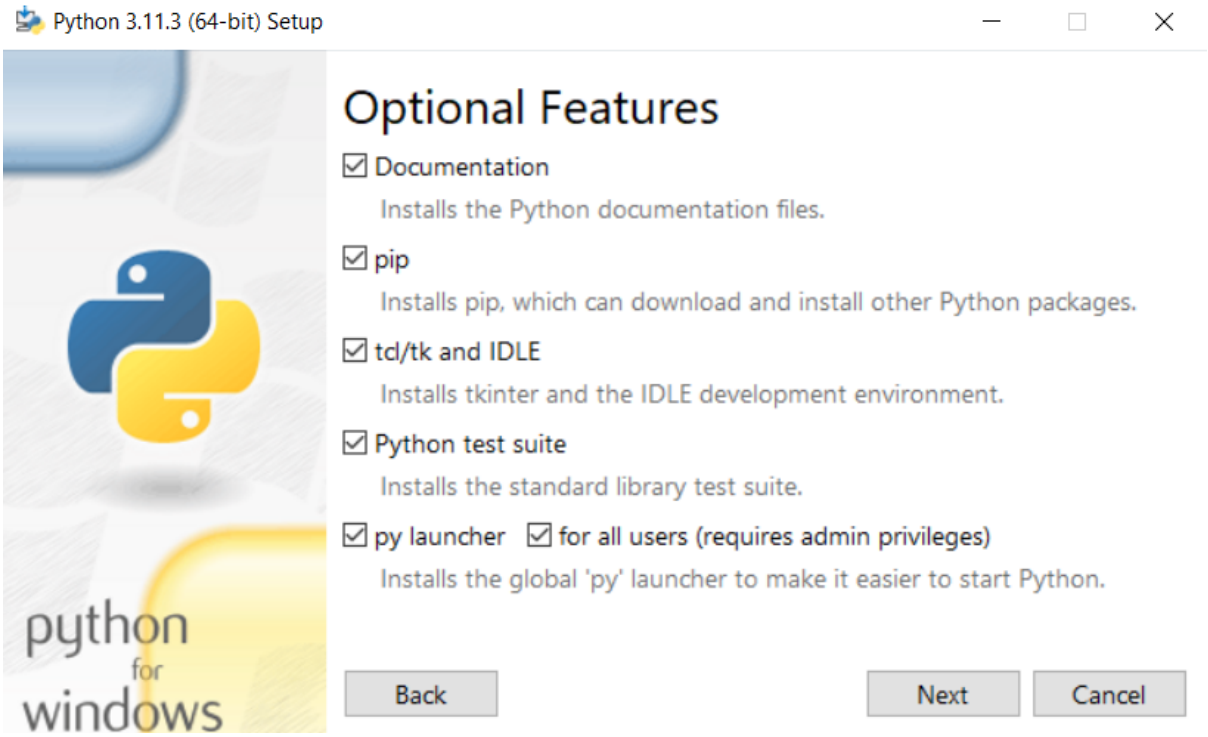
Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		729e36388ae9a832b01cf9138921b383	25007016	SIG
XZ compressed source tarball	Source release		3e7035d272680f80e3ce4e8eb492d580	18726176	SIG
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later (updated for macOS 12 Monterey)	8575cc983035ea2f0414e25ce0289ab8	39735213	SIG
Windows embeddable package (32-bit)	Windows		dc9d1abc644dd78f5e48edae38c7bc6b	7521592	SIG
Windows embeddable package (64-bit)	Windows		340408540eeff359d5eaf93139ab90fd	8474319	SIG
Windows help file	Windows		9d7b80c1c23cfb2cecd63ac4fac9766e	9559706	SIG
Windows installer (32-bit)	Windows		133aa48145032e341ad2a000cd3bff50	27194856	SIG
Windows installer (64-bit)	Windows	Recommended	c3917c08a7fe85db7203da6dcaa99a70	28315928	SIG

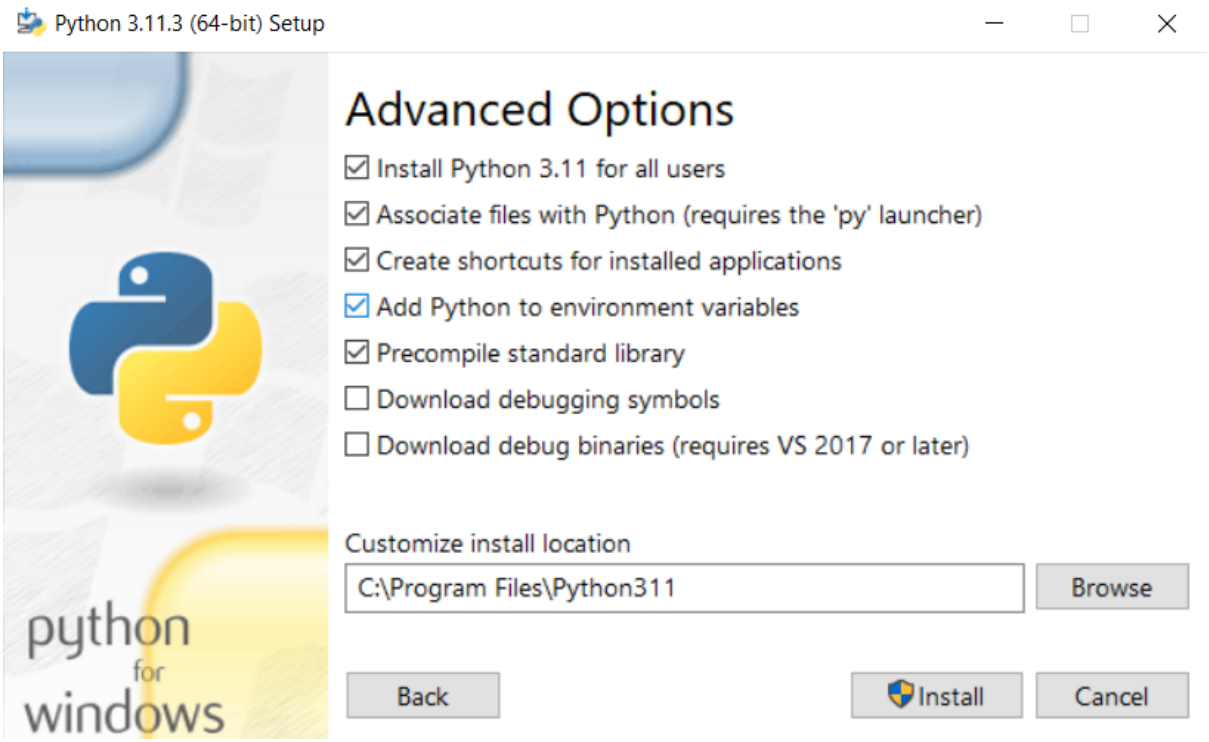
Step 2) Click on the Windows installer (64 bit)

Step 3) Select Customize Installation

Step 4) Click NEXT



Step 5) In next screen
1. Select the advanced options
2. Give a Custom install location. Keep the default folder as c:\Program files\Python311
3. Click Install



Step 6) Click Close button once install is done.

Step 7) Open command prompt window and run the following commands:

- pip install --upgrade pip
- pip install --user -U nltk
- pip install --user -U numpy
- python
- import nltk

```
C:\Users\Admin>python
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> 
```

b) Aim: Convert the given text to speech.

Type these on command prompt to install the packages:

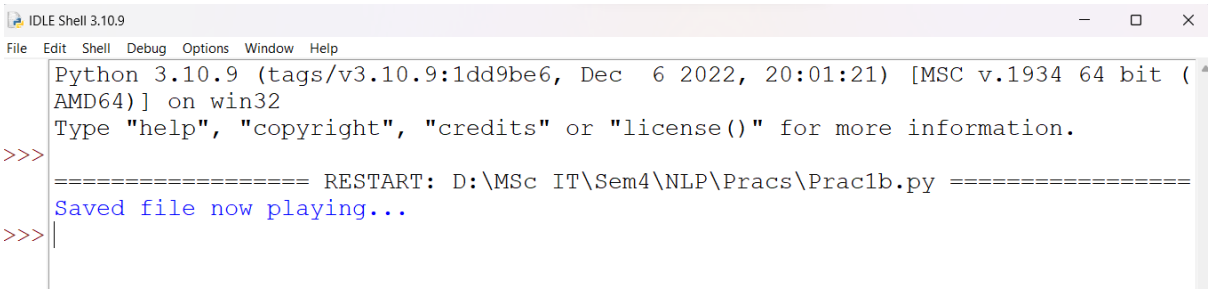
- pip install gtts
- pip install playsound

Source code:

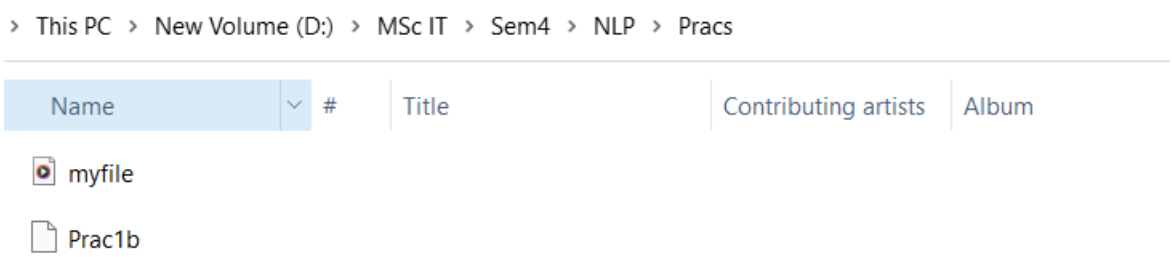
.py file

```
from playsound import playsound

# import required for text to speech conversion

from gtts import gTTS

mytext = "Welcome to Natural Language Programming"

language = "en"

myobj = gTTS(text=mytext, lang=language, slow=False)

myobj.save("myfile.mp3")

print('Saved file now playing...')

playsound("myfile.mp3")
```

Output:



Created file:



c) Aim: Convert audio file Speech to Text.

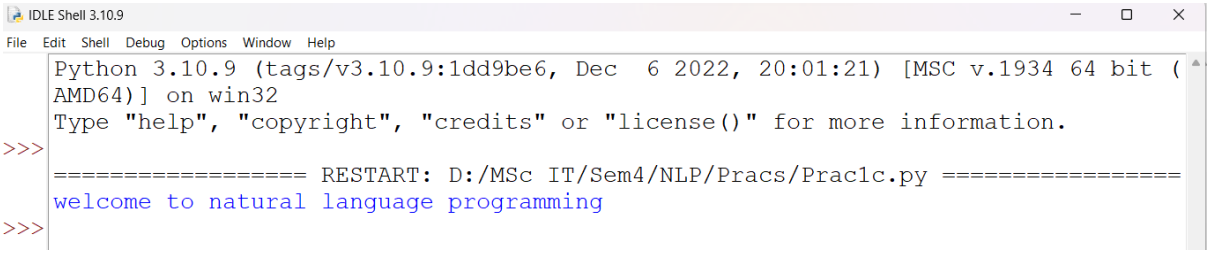
Note: required to store the input file "myfile.wav" in the current folder before running the program.
Type these on command prompt to install the packages:

- pip3 install SpeechRecognition pydub

Source code:

```
import speech_recognition as sr

filename = "myfile1.wav"

# initialize the recognizer

r = sr.Recognizer()

# open the file

with sr.AudioFile(filename) as source:

    # listen for the data (load audio to memory)

    audio_data = r.record(source)

    # recognize (convert from speech to text)

    text = r.recognize_google(audio_data)

    print(text)
```

Output:



Practical Number: 2

a) Aim: Study of various Corpus – Brown, Inaugural, Reuters, udhr with various methods like fields, raw, words, sents, categories**.**

'''NLTK includes a small selection of texts from the Project brown electronic text archive, which contains some 25,000 free electronic books, hosted at http://www.brown.org/. We begin by getting the Python interpreter to load the NLTK package, then ask to see nltk.corpus.brown.fileids(), the file identifiers in this corpus:'''

Source Code:

```
import nltk

from nltk.corpus import brown

print ('File ids of brown corpus\n',brown.fileids())

'''Let's pick out the first of these texts — Emma by Jane Austen — and give it a short

name, emma, then find out how many words it contains:'''

ca01 = brown.words('ca01')

# display first few words

print('\nca01 has following words:\n',ca01)

# total number of words in ca01

print('\nca01 has',len(ca01),'words')

#categories or files

print('\n\nCategories or file in brown corpus:\n')

print(brown.categories())

'''display other information about each text, by looping over all the values of fileid
corresponding to the brown file identifiers listed earlier and then computing statistics for each
text.'''

print('\n\nStatistics for each text:\n')

print('AvgWordLen\tAvgSentenceLen\tNo.ofTimesEachWordAppearsOnAvg\t\tFileName')

for fileid in brown.fileids():

    num_chars = len(brown.raw(fileid))

    num_words = len(brown.words(fileid))

    num_sents = len(brown.sents(fileid))

    num_vocab = len(set([w.lower() for w in brown.words(fileid)]))
```

```
    print(int(num_chars/num_words),'\t\t\t',
int(num_words/num_sents),'\t\t\t',int(num_words/num_vocab),'\t\t\t', fileid)
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac2a.py ====
File ids of brown corpus
  Squeezed text (75 lines).

ca01 has following words:
 ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]

ca01 has 2242 words


Categories or file in brown corpus:

['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies', 'humor', 'learned', 'lore', 'myst
ery', 'news', 'religion', 'reviews', 'romance', 'science_fiction']


Statistics for each text:

AvgWordLen      AvgSentenceLen  No.ofTimesEachWordAppearsOnAvg          FileName
9                       22                      2                        ca01
8                       23                      2                        ca02
8                       20                      2                        ca03
9                       25                      2                        ca04
8                       26                      3                        ca05
8                       22                      2                        ca06
9                       18                      2                        ca07
8                       21                      2                        ca08
9                       19                      2                        ca09
8                       21                      2                        ca10
```

b) Aim: Create and use your own corpora (plaintext, categorical).

Source Code:

```
import nltk
from nltk.corpus import PlaintextCorpusReader
corpus_root = 'D:/MSc IT/Sem4/NLP/Pracs'
filelist = PlaintextCorpusReader(corpus_root, '.*')
print('\n File list: \n')
print(filelist.fileids())
print(filelist.root)
'''display other information about each text, by looping over all the values of fileid
corresponding to the filelist file identifiers listed earlier and then computing statistics for each
text.'''
print('\n\nStatistics for each text:\n')
print('AvgWordLen\tAvgSentenceLen\tNo.ofTimesEachWordAppearsOnAvg\tFileName')
for fileid in filelist.fileids():
    num_chars = len(filelist.raw(fileid))
    num_words = len(filelist.words(fileid))
    num_sents = len(filelist.sents(fileid))
    num_vocab = len(set([w.lower() for w in filelist.words(fileid)]))
    print(int(num_chars/num_words),'\t\t\t',
int(num_words/num_sents),'\t\t\t',int(num_words/num_vocab),'\t\t', fileid)
```

Input folder:

This PC > New Volume (D:) > MSc IT > Sem4 > NLP > Pracs

Name	Date modified	Type	Size
Prac1b	05-04-2023 08:28 PM	Python File	1 KB
Prac1c	05-04-2023 08:29 PM	Python File	1 KB
Prac2a	05-04-2023 08:50 PM	Python File	2 KB
Prac2b	05-04-2023 09:03 PM	Python File	1 KB

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac2b.py =================

 File list:

['Prac1b.py', 'Prac1c.py', 'Prac2a.py', 'Prac2b.py']
D:\MSc IT\Sem4\NLP\Pracs


Statistics for each text:

AvgWordLen      AvgSentenceLen  No.ofTimesEachWordAppearsOnAvg  FileName
4                        67                      1               Prac1b.py
4                        75                      1               Prac1c.py
4                        137                     2               Prac2a.py
4                        66                      2               Prac2b.py
>>>
```

c) Aim: Study Conditional frequency distributions.

Source code:

```
#process a sequence of pairs
text = ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
pairs = [('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ...]
import nltk
from nltk.corpus import brown
fd = nltk.ConditionalFreqDist(
    (genre, word)
    for genre in brown.categories()
    for word in brown.words(categories=genre))
genre_word = [(genre, word)
    for genre in ['news', 'hobbies']
    for word in brown.words(categories=genre)]
print(len(genre_word))
print(genre_word[:4])
print(genre_word[-4:])
cfd = nltk.ConditionalFreqDist(genre_word)
print(cfd)
print(cfd.conditions())
print(cfd['news'])
print(cfd['hobbies'])
print(list(cfd['hobbies']))
from nltk.corpus import inaugural
cfd = nltk.ConditionalFreqDist(
    (target, fileid[:4])
    for fileid in inaugural.fileids()
    for w in inaugural.words(fileid)
    for target in ['america', 'citizen']
    if w.lower().startswith(target))
```

```
from nltk.corpus import udhr
languages = ['Chickasaw', 'English', 'German_Deutsch','Greenlandic_Inuktikut',
'Hungarian_Magyar', 'Ibibio_Efik']
cfd = nltk.ConditionalFreqDist(
    (lang, len(word))
    for lang in languages
    for word in udhr.words(lang + '-Latin1'))
cfd.tabulate(conditions=['English', 'German_Deutsch'],
    samples=range(10), cumulative=True)
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac2c.py =================
182899
[('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ('news', 'Grand')]
[('hobbies', 'the'), ('hobbies', 'Irish'), ('hobbies', 'Sweepstakes'), ('hobbies
', '.')]
<ConditionalFreqDist with 2 conditions>
['news', 'hobbies']
<FreqDist with 14394 samples and 100554 outcomes>
<FreqDist with 11935 samples and 82345 outcomes>
Squeezed text (1688 lines).
                   0    1    2    3    4    5    6    7    8    9
       English     0  185  525  883  997 1166 1283 1440 1558 1638
German_Deutsch     0  171  263  614  717  894 1013 1110 1213 1275
>>>
```

d) Aim: Study of tagged corpora with methods like tagged_sents, tagged_words.

Source Code:

```
import nltk
from nltk import tokenize
nltk.download('punkt')
nltk.download('words')
para = "Hello!We are MSc IT Students. We are doing Natural Language Processing Practicals using Python."
sents = tokenize.sent_tokenize(para)
print("\nSentence tokenization\n===================\n",sents)
# word tokenization
print("\nWord tokenization\n===================\n")
for index in range(len(sents)):
    words = tokenize.word_tokenize(sents[index])
    print(words)
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac2d.py =================

Sentence tokenization
===================
 ['Hello!We are MSc IT Students.', 'We are doing Natural Language Processing Pra
cticals using Python.']

Word tokenization
===================

['Hello', '!', 'We', 'are', 'MSc', 'IT', 'Students', '.']
['We', 'are', 'doing', 'Natural', 'Language', 'Processing', 'Practicals', 'using
', 'Python', '.']
>>>
```

e) Aim: Write a program to find the most frequent noun tags.

Source code:

```
import nltk
from collections import defaultdict
text = nltk.word_tokenize("Nick likes to play football. Nick does not like to play cricket.")
tagged = nltk.pos_tag(text)
print(tagged)
# checking if it is a noun or not
addNounWords = []
count=0
for words in tagged:
    val = tagged[count][1]
    if(val == 'NN' or val == 'NNS' or val == 'NNPS' or val == 'NNP'):
        addNounWords.append(tagged[count][0])
    count+=1
print(addNounWords)
temp = defaultdict(int)
# memoizing count
for sub in addNounWords:
    for wrd in sub.split():
        temp[wrd] += 1
# getting max frequency
res = max(temp, key=temp.get)
print("Word with maximum frequency : " + str(res))  # printing result
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac2e.py =================
[('Nick', 'NNP'), ('likes', 'VBZ'), ('to', 'TO'), ('play', 'VB'), ('football', 'NN'), ('.', '.'), ('Nick', 'NNP'),
('does', 'VBZ'), ('not', 'RB'), ('like', 'VB'), ('to', 'TO'), ('play', 'VB'), ('cricket', 'NN'), ('.', '.')]
['Nick', 'football', 'Nick', 'cricket']
Word with maximum frequency : Nick
>>>
```

f) Aim: Map Words to Properties Using Python Dictionaries.

Source Code:

```
#creating and printing a dictionay by mapping word with its properties
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
}
print(thisdict)
print(thisdict["brand"])
print(len(thisdict))
print(type(thisdict))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
    Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
    AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more information.
>>>
    ================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac2f.py ==================
    {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
    Ford
    3
    <class 'dict'>
>>>
```

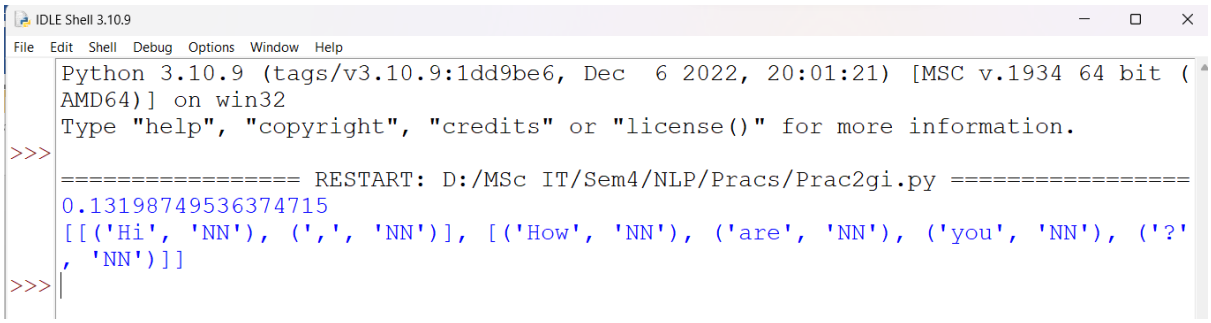
g) Aim: Study i) DefaultTagger, ii) Regular expression tagger, iii) UnigramTagger.

i) DefaultTagger

Source Code:

```
import warnings
warnings.filterwarnings("ignore")
import nltk
from nltk.tag import DefaultTagger
exptagger = DefaultTagger('NN')
from nltk.corpus import treebank
testsentences = treebank.tagged_sents() [1000:]
print(exptagger.evaluate (testsentences))
#Tagging a list of sentences
import nltk
from nltk.tag import DefaultTagger
exptagger = DefaultTagger('NN')
print(exptagger.tag_sents([['Hi', ','], ['How', 'are', 'you', '?']]))
```

Output:



ii) Regular expression tagger

Source code:

```
from nltk.corpus import brown
from nltk.tag import RegexpTagger
test_sent = brown.sents(categories='news')[0]
regexp_tagger = RegexpTagger(
    [(r'^-?[0-9]+(.[0-9]+)?$', 'CD'), # cardinal numbers
    (r'(The|the|A|a|An|an)$', 'AT'), # articles
    (r'.*able$', 'JJ'), # adjectives
    (r'.*ness$', 'NN'), # nouns formed from adjectives
    (r'.*ly$', 'RB'), # adverbs
    (r'.*s$', 'NNS'), # plural nouns
    (r'.*ing$', 'VBG'), # gerunds
    (r'.*ed$', 'VBD'), # past tense verbs
    (r'.*', 'NN') # nouns (default)
])
print(regexp_tagger)
print(regexp_tagger.tag(test_sent))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac2gii.py ================
<Regexp Tagger: size=9>
[('The', 'AT'), ('Fulton', 'NN'), ('County', 'NN'), ('Grand', 'NN'), ('Jury', 'NN'), ('said', 'NN'), ('Friday', 'N
N'), ('an', 'AT'), ('investigation', 'NN'), ('of', 'NN'), ("Atlanta's", 'NNS'), ('recent', 'NN'), ('primary', 'NN'
), ('election', 'NN'), ('produced', 'VBD'), ('``', 'NN'), ('no', 'NN'), ('evidence', 'NN'), ("''", 'NN'), ('that',
'NN'), ('any', 'NN'), ('irregularities', 'NNS'), ('took', 'NN'), ('place', 'NN'), ('.', 'NN')]
>>>
```

iii) Unigram Tagger

Source Code:

```
# Loading Libraries
from nltk.tag import UnigramTagger
from nltk.corpus import treebank
# Training using first 10 tagged sentences of the treebank corpus as data.
# Using data
train_sents = treebank.tagged_sents()[:10]
# Initializing
tagger = UnigramTagger(train_sents)
# Lets see the first sentence
# (of the treebank corpus) as list
print(treebank.sents()[0])
print('\n',tagger.tag(treebank.sents()[0]))
#Finding the tagged results after training.
tagger.tag(treebank.sents()[0])
#Overriding the context model
tagger = UnigramTagger(model ={'Pierre': 'NN'})
print('\n',tagger.tag(treebank.sents()[0]))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac2giii.py ================
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the', 'boa
rd', 'as', 'a', 'nonexecutive', 'director', 'Nov.', '29', '.']

 [('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS
'), ('old', 'JJ'), (',', ','), ('will', 'MD'), ('join', 'VB'), ('the', 'DT'), ('
board', 'NN'), ('as', 'IN'), ('a', 'DT'), ('nonexecutive', 'JJ'), ('director', '
NN'), ('Nov.', 'NNP'), ('29', 'CD'), ('.', '.')]

 [('Pierre', 'NN'), ('Vinken', None), (',', None), ('61', None), ('years', None)
, ('old', None), (',', None), ('will', None), ('join', None), ('the', None), ('b
oard', None), ('as', None), ('a', None), ('nonexecutive', None), ('director', No
ne), ('Nov.', None), ('29', None), ('.', None)]
>>>
```

h. Find different words from a given plain text without any space by comparing this text with a given corpus of words. Also find the score of words.

Question:

Initialize the hash tag test data or URL test data and convert to plain text without any space. Read a text file of different words and compare the plain text data with the words exist in that text file and find out different words available in that plain text. Also find out how many words could be found. (for example, text = "#whatismyname" or text = www.whatismyname.com. Convert that to plain text without space as: whatismyname and read text file as words.txt. Now compare plain text with words given in a file and find the words form the plain text and the count of words which could be found)

Source code:

```
from __future__ import with_statement #with statement for reading file

import re # Regular expression

words = [] # corpus file words

testword = [] # test words

ans = [] # words matches with corpus

print("MENU")

print("-----------")

print(" 1 . Hash tag segmentation ")

print(" 2 . URL segmentation ")

print("Enter the input choice for performing word segmentation")

choice = int(input())

if choice == 1:

    text = "#whatismyname" # hash tag test data to segment

    print("Input with HashTag",text)

    pattern=re.compile("[^\w']")

    a = pattern.sub(", text)

elif choice == 2:

    text = "www.whatismyname.com" # url test data to segment

    print("Input with URL",text)

    a=re.split('\s|(?<!\d)[,.](?!\d)', text)

    splitwords = ["www","com","in"] # remove the words which is containg in the list
```

```
    a ="".join([each for each in a if each not in splitwords])
else:
    print("Wrong choice...Try again")
print(a)
for each in a:
    testword.append(each) #test word
test_lenth = len(testword) # lenth of the test data
# Reading the corpus
with open('words1.txt', 'r') as f:
    lines = f.readlines()
    words =[(e.strip()) for e in lines]
def Seg(a,lenth):
    ans =[]
    for k in range(0,lenth+1): # this loop checks char by char in the corpus
        if a[0:k] in words:
            print(a[0:k],"-appears in the corpus")
            ans.append(a[0:k])
            break
    if ans != []:
        g = max(ans,key=len)
        return g
test_tot_itr = 0 #each iteration value
answer = [] # Store the each word contains the corpus
Score = 0 # initial value for score
N = 37 # total no of corpus
M = 0
C = 0
while test_tot_itr < test_lenth:
    ans_words = Seg(a,test_lenth)
    if ans_words != 0:
        test_itr = len(ans_words)
        answer.append(ans_words)
```

```
        a = a[test_itr:test_lenth]
        test_tot_itr += test_itr
Aft_Seg = " ".join([each for each in answer])
# print segmented words in the list
print("Output")
print("---------")
print(Aft_Seg) # print After segmentation the input
# Calculating Score
C = len(answer)
score = C * N / N # Calculate the score
print("Score",score)
```

Input file:

words1

File Edit View

```
check
back
domain
social
big
media
rocks
30
name
seconds
cheap
earth
being
this
human
is
current
insane
rates
it
ought
time
to
what
go
is
down
my
apple
name
domains
let
honesty
us
hour
go
follow
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac2h.py =================
MENU
-----------
 1 . Hash tag segmentation
 2 . URL segmentation
Enter the input choice for performing word segmentation
1
Input with HashTag #whatismyname
whatismyname
what -appears in the corpus
is -appears in the corpus
my -appears in the corpus
name -appears in the corpus
Output
---------
what is my name
Score 4.0
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac2h.py =================
MENU
-----------
 1 . Hash tag segmentation
 2 . URL segmentation
Enter the input choice for performing word segmentation
2
Input with URL www.whatismyname.com
whatismyname
what -appears in the corpus
is -appears in the corpus
my -appears in the corpus
name -appears in the corpus
Output
---------
what is my name
Score 4.0
>>>
```

Practical Number: 3

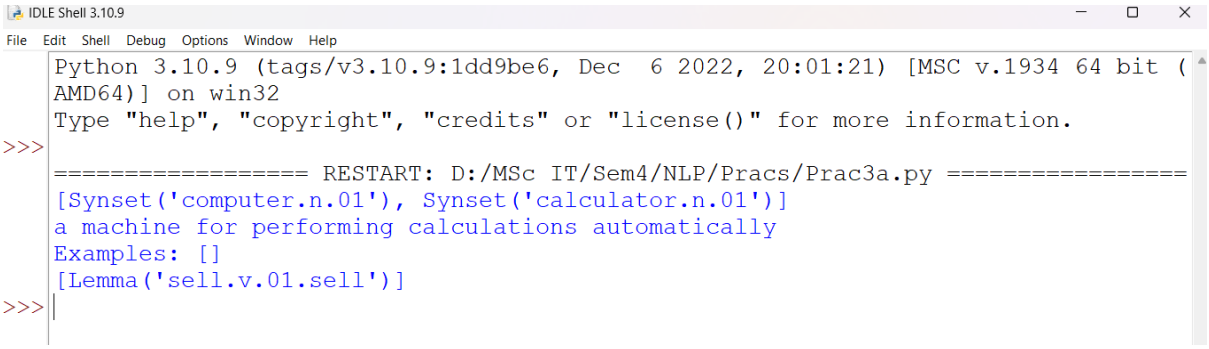
a) Aim: Study of Wordnet Dictionary with methods as synsets, definitions, examples, antonyms.

WordNet provides synsets which is the collection of synonym words also called “lemmas”

Source Code:

```
import nltk

from nltk.corpus import wordnet

print(wordnet.synsets("computer"))

# definition and example of the word ‘computer’

print(wordnet.synset("computer.n.01").definition())

#examples

print("Examples:", wordnet.synset("computer.n.01").examples())

#get Antonyms

print(wordnet.lemma('buy.v.01.buy').antonyms())
```

Output:



b) Aim: Study lemmas, hyponyms, hypernyms.

Source code:

```
import nltk
from nltk.corpus import wordnet
print(wordnet.synsets("computer"))
print(wordnet.synset("computer.n.01").lemma_names())
#all lemmas for each synset.
for e in wordnet.synsets("computer"):
    print(f'{e} --> {e.lemma_names()}')
#print all lemmas for a given synset
print(wordnet.synset('computer.n.01').lemmas())
#get the synset corresponding to lemma
print(wordnet.lemma('computer.n.01.computing_device').synset())
#Get the name of the lemma
print(wordnet.lemma('computer.n.01.computing_device').name())
#Hyponyms give abstract concepts of the word that are much more specific
#the list of hyponyms words of the computer
syn = wordnet.synset('computer.n.01')
print(syn.hyponyms)
print([lemma.name() for synset in syn.hyponyms() for lemma in synset.lemmas()])
#the semantic similarity in WordNet
vehicle = wordnet.synset('vehicle.n.01')
car = wordnet.synset('car.n.01')
print(car.lowest_common_hypernyms(vehicle))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac3b.py =================
[Synset('computer.n.01'), Synset('calculator.n.01')]
['computer', 'computing_machine', 'computing_device', 'data_processor', 'electronic_computer', 'information_proces
sing_system']
Synset('computer.n.01') --> ['computer', 'computing_machine', 'computing_device', 'data_processor', 'electronic_co
mputer', 'information_processing_system']
Synset('calculator.n.01') --> ['calculator', 'reckoner', 'figurer', 'estimator', 'computer']
[Lemma('computer.n.01.computer'), Lemma('computer.n.01.computing_machine'), Lemma('computer.n.01.computing_device'
), Lemma('computer.n.01.data_processor'), Lemma('computer.n.01.electronic_computer'), Lemma('computer.n.01.informa
tion_processing_system')]
Synset('computer.n.01')
computing_device
<bound method _WordNetObject.hyponyms of Synset('computer.n.01')>
['analog_computer', 'analogue_computer', 'digital_computer', 'home_computer', 'node', 'client', 'guest', 'number_c
runcher', 'pari-mutuel_machine', 'totalizer', 'totaliser', 'totalizator', 'totalisator', 'predictor', 'server', 'h
ost', 'Turing_machine', 'web_site', 'website', 'internet_site', 'site']
[Synset('vehicle.n.01')]
>>>
```

c) Aim: Write a program using python to find synonym and antonym of word "active" using Wordnet.

Source Code:

```
from nltk.corpus import wordnet

print( wordnet.synsets("active"))

print(wordnet.lemma('active.a.01.active').antonyms())
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac3c.py =================
[Synset('active_agent.n.01'), Synset('active_voice.n.01'), Synset('active.n.03'), Synset('active.a.01'), Synset('a
ctive.s.02'), Synset('active.a.03'), Synset('active.s.04'), Synset('active.a.05'), Synset('active.a.06'), Synset('
active.a.07'), Synset('active.s.08'), Synset('active.a.09'), Synset('active.a.10'), Synset('active.a.11'), Synset(
'active.a.12'), Synset('active.a.13'), Synset('active.a.14')]
[Lemma('inactive.a.02.inactive')]
>>>
```

d) Aim: Compare two nouns.

Source Code:

```
import nltk
from nltk.corpus import wordnet
syn1 = wordnet.synsets('football')
syn2 = wordnet.synsets('soccer')
# A word may have multiple synsets, so need to compare each synset of word1 with synset of word2
for s1 in syn1:
    for s2 in syn2:
        print("Path similarity of: ")
        print(s1, '(', s1.pos(), ')', '[', s1.definition(), ']')
        print(s2, '(', s2.pos(), ')', '[', s2.definition(), ']')
        print(" is", s1.path_similarity(s2))
        print()
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac3d.py ====
Path similarity of:
Synset('football.n.01') ( n ) [ any of various games played with a ball (round or oval) in which two teams try to kick or carry or propel the ball into each other's goal ]
Synset('soccer.n.01') ( n ) [ a football game in which two teams of 11 players try to kick or head a ball into the opponents' goal ]
 is 0.5

Path similarity of:
Synset('football.n.02') ( n ) [ the inflated oblong ball used in playing American football ]
Synset('soccer.n.01') ( n ) [ a football game in which two teams of 11 players try to kick or head a ball into the opponents' goal ]
 is 0.05

>>>
```

e) Aim: Handling stopword.

i) Using nltk Adding or Removing Stop Words in NLTK's Default Stop Word List

Source Code:

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.tokenize import word_tokenize
text = "Yashesh likes to play football, however he is not too fond of tennis."
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()]
print(tokens_without_sw)
#add the word play to the NLTK stop word collection
all_stopwords = stopwords.words('english')
all_stopwords.append('play')
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]
print(tokens_without_sw)
#remove 'not' from stop word collection
all_stopwords.remove('not')
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]
print(tokens_without_sw)
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac3e.py =================
[nltk_data] Downloading package stopwords to C:\Users\Prerana P
[nltk_data]     Byahatti\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
['Yashesh', 'likes', 'play', 'football', ',', 'fond', 'tennis', '.']
['Yashesh', 'likes', 'football', ',', 'however', 'fond', 'tennis', '.']
['Yashesh', 'likes', 'football', ',', 'however', 'not', 'fond', 'tennis', '.']
>>>
```

ii) Using Gensim Adding and Removing Stop Words in Default Gensim Stop Words List

Type these on command prompt to install the package:

- pip3 install genism

Source Code:

```
import gensim

from gensim.parsing.preprocessing import remove_stopwords

from nltk.tokenize import word_tokenize

text = "Yashesh likes to play football, however he is not too fond of tennis."

filtered_sentence = remove_stopwords(text)

print(filtered_sentence)

all_stopwords = gensim.parsing.preprocessing.STOPWORDS

#print(all_stopwords)

'''The following script adds likes and play to the list of stop words in Gensim:'''

from gensim.parsing.preprocessing import STOPWORDS

all_stopwords_gensim = STOPWORDS.union(set(['likes', 'play']))

text = "Yashesh likes to play football, however he is not too fond of tennis."

text_tokens = word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in all_stopwords_gensim]

#print(tokens_without_sw)

from gensim.parsing.preprocessing import STOPWORDS

all_stopwords_gensim = STOPWORDS

sw_list = {"not"}

all_stopwords_gensim = STOPWORDS.difference(sw_list)

text = "Yashesh likes to play football, however he is not too fond of tennis."

text_tokens = word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in all_stopwords_gensim]

print(tokens_without_sw)
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac3eii.py ================
Yashesh likes play football, fond tennis.
['Yashesh', 'likes', 'play', 'football', ',', 'not', 'fond', 'tennis', '.']
>>>
```

iii) Using Spacy Adding and Removing Stop Words in Default Spacy Stop Words List

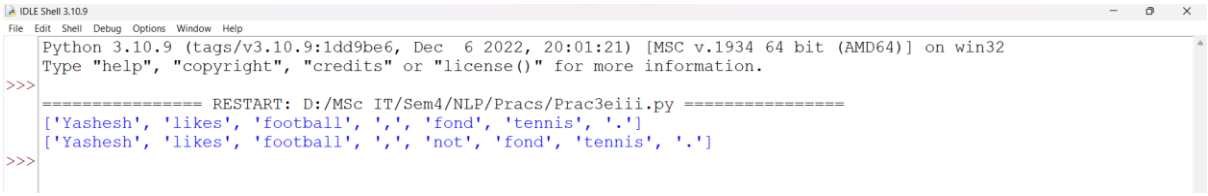
Type these on command prompt to install the packages:

- pip3 install spacy
- python -m spacy download en_core_web_sm
- python -m spacy download en

Source Code:

```
import spacy

import nltk

from nltk.tokenize import word_tokenize

sp = spacy.load('en_core_web_sm')

#add the word play to the NLTK stop word collection

all_stopwords = sp.Defaults.stop_words

all_stopwords.add("play")

text = "Yashesh likes to play football, however he is not too fond of tennis."

text_tokens = word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]

print(tokens_without_sw)

#remove 'not' from stop word collection

all_stopwords.remove('not')

tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]

print(tokens_without_sw)
```

Output:



Practical Number: 4

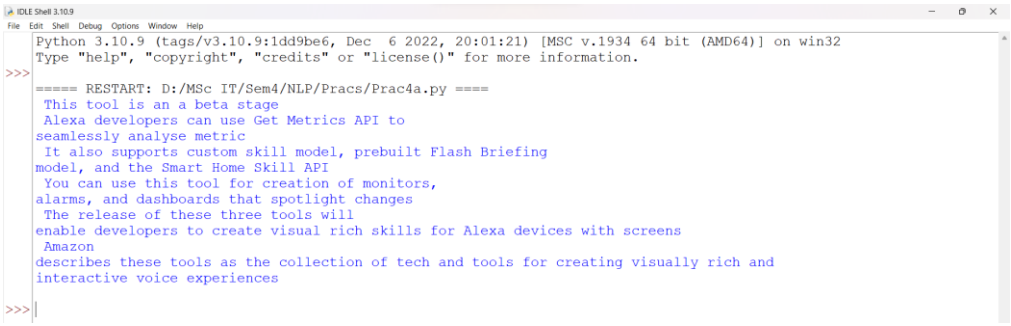
Text Tokenization

a) **Aim:** Tokenization using Python's split() function.

Source Code:

```
text = """ This tool is an a beta stage. Alexa developers can use Get Metrics API to seamlessly analyse metric. It also supports custom skill model, prebuilt Flash Briefing model, and the Smart Home Skill API. You can use this tool for creation of monitors, alarms, and dashboards that spotlight changes. The release of these three tools will enable developers to create visual rich skills for Alexa devices with screens. Amazon describes these tools as the collection of tech and tools for creating visually rich and interactive voice experiences. """
data = text.split('.')
for i in data:
    print(i)
```

Output:



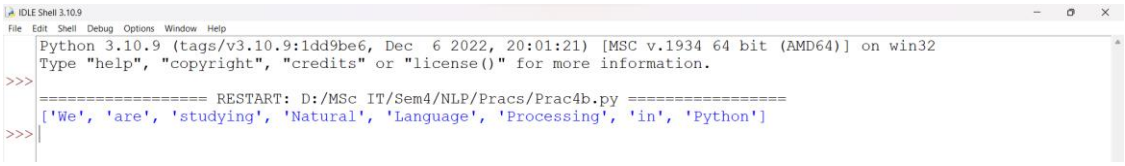
```
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac4a.py ====
 This tool is an a beta stage
 Alexa developers can use Get Metrics API to
seamlessly analyse metric
 It also supports custom skill model, prebuilt Flash Briefing
model, and the Smart Home Skill API
 You can use this tool for creation of monitors,
alarms, and dashboards that spotlight changes
 The release of these three tools will
enable developers to create visual rich skills for Alexa devices with screens
 Amazon
describes these tools as the collection of tech and tools for creating visually rich and
interactive voice experiences
>>>
```

b) Aim: Tokenization using Regular Expressions (RegEx).

Source Code:

```
import nltk
# import RegexpTokenizer() method from nltk
from nltk.tokenize import RegexpTokenizer
# Create a reference variable for Class RegexpTokenizer
tk = RegexpTokenizer('\s+', gaps = True)
# Create a string input
str = "We are studying Natural Language Processing in Python"
# Use tokenize method
tokens = tk.tokenize(str)
print(tokens)
```

Output:



c) **Aim:** Tokenization using NLTK.

Source Code:

```
import nltk
from nltk.tokenize import word_tokenize
# Create a string input
str = "We are studying Natural Language Processing in Python"
# Use tokenize method
print(word_tokenize(str))
```

Output:

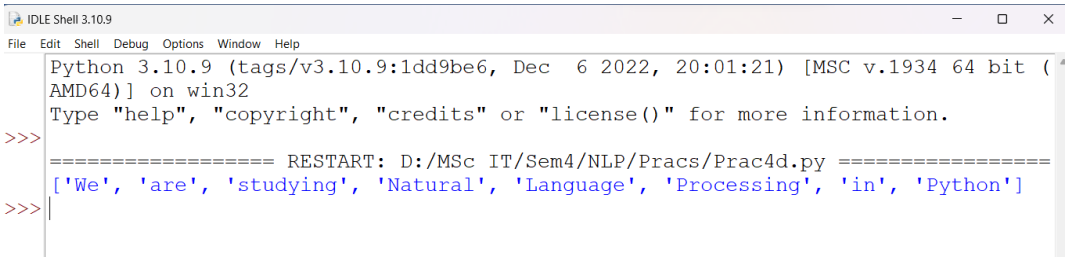
```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac4c.py =================
['We', 'are', 'studying', 'Natural', 'Language', 'Processing', 'in', 'Python']
>>>
```

d) Aim: Tokenization using the spaCy library.

Source Code:

```
import spacy
nlp = spacy.blank("en")
# Create a string input
str = "We are studying Natural Language Processing in Python"
# Create an instance of document;
# doc object is a container for a sequence of Token objects.
doc = nlp(str)
# Read the words; Print the words
words = [word.text for word in doc]
print(words)
```

Output:



e) **Aim:** Tokenization using Keras.

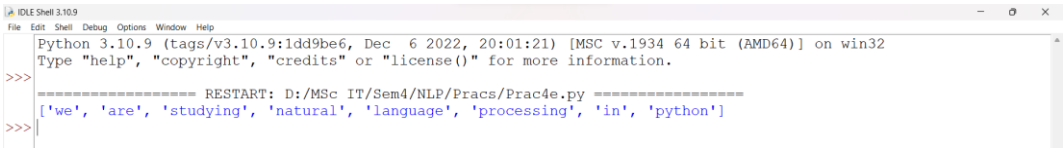
Type these on command prompt to install the packages:

- pip install keras
- pip install tensorflow

Source Code:

```
import keras

from keras.preprocessing.text import text_to_word_sequence

# Create a string input

str = "We are studying Natural Language Processing in Python"

# tokenizing the text

tokens = text_to_word_sequence(str)

print(tokens)
```

Output:



f) Aim: Tokenization using Gensim.

Type these on command prompt to install the packages:

- pip install genism

Source Code:

```
from gensim.utils import tokenize

# Create a string input

str = "We are studying Natural Language Processing in Python"

# tokenizing the text

list(tokenize(str))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac4f.py =================
['We', 'are', 'studying', 'Natural', 'Language', 'Processing', 'in', 'Python']
>>>
```

Practical Number: 5

Import NLP Libraries for Indian Languages and perform:

a) Aim: Word tokenization in Hindi.

Source Code:

```
import nltk
from indicnlp.tokenize import indic_tokenize
# Set the Hindi language as the default language for tokenization
nltk.download('punkt')
nltk.download('indian')
#nltk.data.path.append("/path/to/indic_nlp_library/indic_nlp_resources")
# Tokenize a Hindi sentence
sentence = "विभिन्न भाषाओं को सीखना दिलचस्प है।"
tokens = indic_tokenize.trivial_tokenize(sentence)
# Print the tokens
print(tokens)
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:\MSc IT\Sem4\NLP\Pracs\Prac5a.py =================
[nltk_data] Downloading package punkt to C:\Users\Prerana P
[nltk_data]     Byahatti\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package indian to C:\Users\Prerana P
[nltk_data]     Byahatti\AppData\Roaming\nltk_data...
[nltk_data]   Package indian is already up-to-date!
['विभिन्न', 'भाषाओं', 'को', 'सीखना', 'दिलचस्प', 'है', '।']
>>>
```

b) Aim: Generate similar sentences from a given Hindi text input.

Source code:

Note: Execute this practical in https://colab.research.google.com/

```
!pip install torch==1.3.1+cpu -f https://download.pytorch.org/whl/torch_stable.html
!pip install inltk
!pip install tornado==4.5.3
from inltk.inltk import setup
setup('hi')
from inltk.inltk import get_similar_sentences
# get similar sentences to the one given in hindi
output = get_similar_sentences('मैं आज बहुि खुश हूं', 5, 'hi')
print(output)
```

Output:

['मैं आजकल बहुत खुश हूं', 'मैं आज अत्यधिक खुश हूं', 'मैं अभी बहुत खुश हूं', 'मैं वर्तमान बहुत खुश हूं', 'मैं वर्तमान बहुत खुश हूं']

c) <u>Aim:</u> Identify the Indian language of a text.

<u>Source Code:</u>

```
import langid
# Define a text in an Indian language
text = "બીના કાપડિયા"
# Identify the language of the text
lang, conf = langid.classify(text)
# Print the language and the confidence score
print("Language and confidence score of ",text," is ",lang,",", conf)
```

<u>Output</u>:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:\MSc IT\Sem4\NLP\Pracs\Prac5c.py =================
Language and confidence score of  બીના કાપડિયા  is  gu , -72.40245139598846
>>>
```

Practical Number: 6

Illustrate part of speech tagging.

a) **Aim:** Part of speech Tagging and chunking of user defined text.

Source Code:

```
import nltk
from nltk import tokenize
nltk.download('punkt')
from nltk import tag
from nltk import chunk
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
para = "Hello! We are students from MSc IT. Today we are learning NLTK."
sents = tokenize.sent_tokenize(para)
print("\nSentence tokenization\n===================\n",sents)
# word tokenization
print("\nWord tokenization\n===================\n")
for index in range(len(sents)):
    words = tokenize.word_tokenize(sents[index])
    print(words)
# POS Tagging
tagged_words = []
for index in range(len(sents)):
    tagged_words.append(tag.pos_tag(words))
print("\nPOS Tagging\n===========\n",tagged_words)
# chunking
tree = []
for index in range(len(sents)):
    tree.append(chunk.ne_chunk(tagged_words[index]))
```

```
print("\nChunking\n========\n")
print(tree)
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac6a.py =================
[nltk_data] Downloading package punkt to C:\Users\Prerana P
[nltk_data]     Byahatti\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Prerana P
[nltk_data]     Byahatti\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to C:\Users\Prerana
[nltk_data]     P Byahatti\AppData\Roaming\nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to C:\Users\Prerana P
[nltk_data]     Byahatti\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!

Sentence tokenization
=====================
 ['Hello!', 'We are students from MSc IT.', 'Today we are learning NLTK.']

Word tokenization
=====================

['Hello', '!']
['We', 'are', 'students', 'from', 'MSc', 'IT', '.']
['Today', 'we', 'are', 'learning', 'NLTK', '.']

POS Tagging
===========
 [[('Today', 'NN'), ('we', 'PRP'), ('are', 'VBP'), ('learning', 'VBG'), ('NLTK', 'NNP'), ('.', '.')], [('Today', 'NN'), ('we', 'PRP'), ('are', 'VBP'), ('
learning', 'VBG'), ('NLTK', 'NNP'), ('.', '.')], [('Today', 'NN'), ('we', 'PRP'), ('are', 'VBP'), ('learning', 'VBG'), ('NLTK', 'NNP'), ('.', '.')]]

Chunking
========

[Tree('S', [('Today', 'NN'), ('we', 'PRP'), ('are', 'VBP'), ('learning', 'VBG'), Tree('ORGANIZATION', [('NLTK', 'NNP')]), ('.', '.')]), Tree('S', [('Toda
y', 'NN'), ('we', 'PRP'), ('are', 'VBP'), ('learning', 'VBG'), Tree('ORGANIZATION', [('NLTK', 'NNP')]), ('.', '.')]), Tree('S', [('Today', 'NN'), ('we',
'PRP'), ('are', 'VBP'), ('learning', 'VBG'), Tree('ORGANIZATION', [('NLTK', 'NNP')]), ('.', '.')])]
>>>
```

b) Aim: Named Entity recognition using user defined text.

Type these on command prompt to install the packages:

- pip3 install -U spacy
- python -m spacy download en_core_web_sm

Source code:

```
import spacy
# Load English tokenizer, tagger, parser and NER
nlp = spacy.load("en_core_web_sm")
# Process whole documents
text = ("When Sebastian Thrun started working on self-driving cars at "
"Google in 2007, few people outside of the company took him "
"seriously. “I can tell you very senior CEOs of major American "
"car companies would shake my hand and turn away because I wasn’t "
"worth talking to,” said Thrun, in an interview with Recode earlier "
"this week.")
doc = nlp(text)
# Analyse syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])
```

Output:

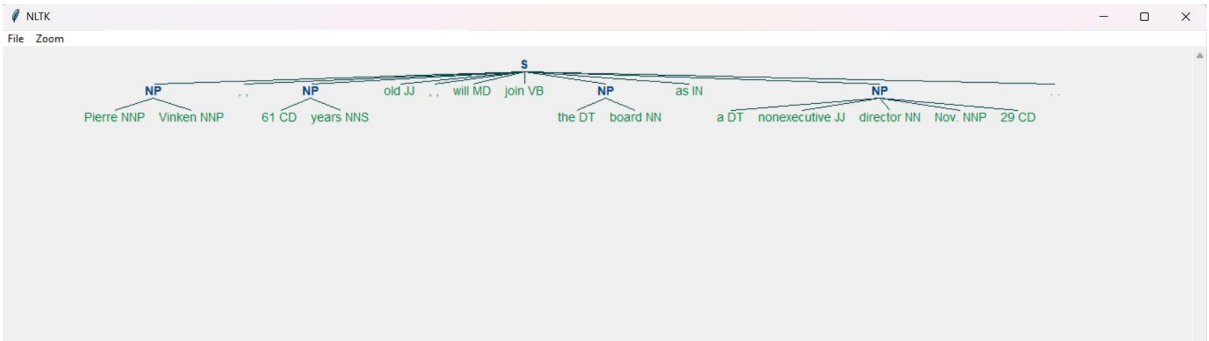
```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac6b.py =================
Noun phrases: ['Sebastian Thrun', 'self-driving cars', 'Google', 'few people', 'the company', 'him', 'I', 'you', '
very senior CEOs', 'major American car companies', 'my hand', 'I', 'Thrun', 'an interview', 'Recode']
Verbs: ['start', 'work', 'drive', 'take', 'tell', 'shake', 'turn', 'talk', 'say']
>>>
```

c) Aim: Named Entity recognition with diagram using NLTK corpus – treebank.

Source code:

```
import nltk
nltk.download('treebank')
from nltk.corpus import treebank_chunk
treebank_chunk.tagged_sents()[0]
treebank_chunk.chunked_sents()[0]
treebank_chunk.chunked_sents()[0].draw()
```

Output:



Practical Number: 7

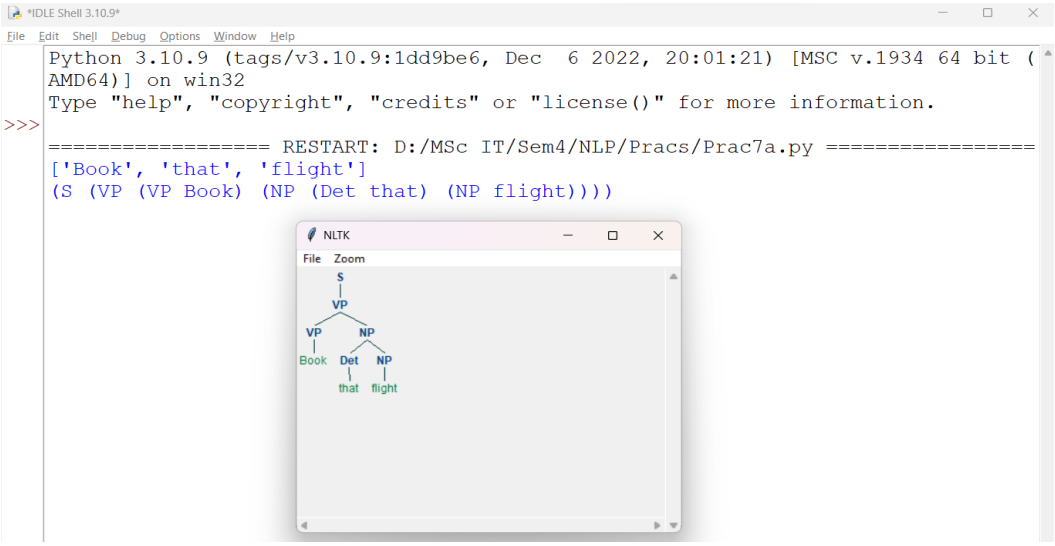
Finite state automata

a) Aim: Define grammar using nltk. Analyze a sentence using the same.

Source Code:

```
import nltk
from nltk import tokenize
grammar1 = nltk.CFG.fromstring("""
   S -> VP
   VP -> VP NP
   NP -> Det NP
   Det -> 'that'
   NP -> singular Noun
   NP -> 'flight'
   VP -> 'Book'
""")
sentence = "Book that flight"
for index in range(len(sentence)):
   all_tokens = tokenize.word_tokenize(sentence)
print(all_tokens)
parser = nltk.ChartParser(grammar1)
for tree in parser.parse(all_tokens):
   print(tree)
   tree.draw()
```

Output:


IDLE Shell 3.10.9
File Edit Shell Debug Options Window Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec 6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac7a.py ==========
['Book', 'that', 'flight']
(S (VP (VP Book) (NP (Det that) (NP flight))))
NLTK
File Zoom
S
VP
VP
NP
Book
Det
NP
that
flight

b) Aim: Accept the input string with Regular expression of Finite Automaton: 101+.

Source code:

```
def FA(s):
#if the length is less than 3 then it can't be accepted, Therefore end the process.
    if len(s)<3:
        return "Rejected"
#first three characters are fixed. Therefore, checking them using index
    if s[0]=='1':
        if s[1]=='0':
            if s[2]=='1':
                # After index 2 only "1" can appear. Therefore break the process if any
othercharacter is detected
                for i in range(3,len(s)):
                    if s[i]!='1':
                        return "Rejected"
                return "Accepted" # if all 4 nested if true
            return "Rejected" # else of 3rd if
        return "Rejected" # else of 2nd if
    return "Rejected" # else of 1st if
inputs=['1','10101','101','10111','01010','100','','10111101','1011111']
for i in inputs:
    print(FA(i))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac7b.py ================
Rejected
Rejected
Accepted
Accepted
Rejected
Rejected
Rejected
Rejected
Accepted
>>>
```

c) Aim: Accept the input string with Regular expression of FA: (a+b)*bba.

Source Code:

```
def FA(s):
    size=0
#scan complete string and make sure that it contains only 'a' & 'b'
    for i in s:
        if i=='a' or i=='b':
            size+=1
        else:
            return "Rejected"
#After checking that it contains only 'a' & 'b'
#check it's length it should be 3 atleast
    if size>=3:
#check the last 3 elements
        if s[size-3]=='b':
            if s[size-2]=='b':
                if s[size-1]=='a':
                    return "Accepted" # if all 4 if true
                return "Rejected" # else of 4th if
            return "Rejected" # else of 3rd if
        return "Rejected" # else of 2nd if
    return "Rejected" # else of 1st if
inputs=['bba','ababbba','abba','abb','baba','bbb','']
for i in inputs:
    print(FA(i))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac7c.py =================
Accepted
Accepted
Accepted
Rejected
Rejected
Rejected
Rejected
>>>
```

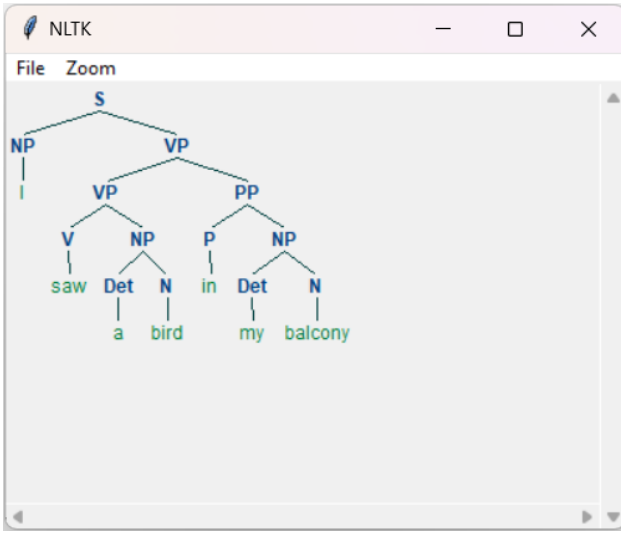
d) Aim; Implementation of Deductive Chart Parsing using context free grammar and a given sentence.

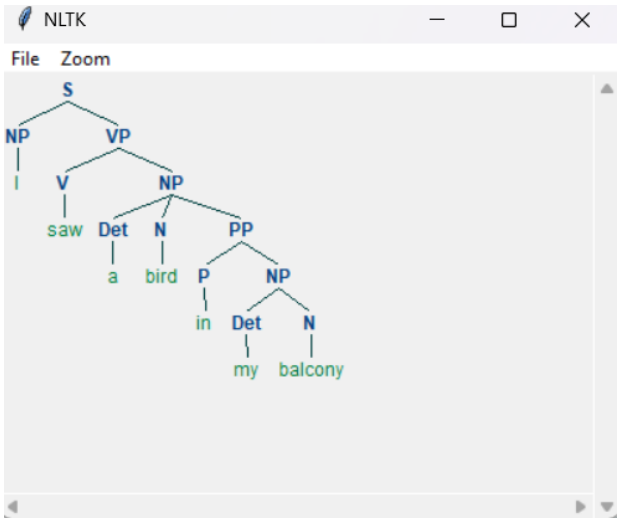
Source code:

```
import nltk
from nltk import tokenize
grammar1 = nltk.CFG.fromstring("""
   S -> NP VP
   PP -> P NP
   NP -> Det N | Det N PP | 'I'
   VP -> V NP | VP PP
   Det -> 'a' | 'my'
   N -> 'bird' | 'balcony'
   V -> 'saw'
   P -> 'in'
""")
sentence = "I saw a bird in my balcony"
for index in range(len(sentence)):
   all_tokens = tokenize.word_tokenize(sentence)
print(all_tokens)
# all_tokens = ['I', 'saw', 'a', 'bird', 'in', 'my', 'balcony']
parser = nltk.ChartParser(grammar1)
for tree in parser.parse(all_tokens):
   print(tree)
   tree.draw()
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac7d.py =================
['I', 'saw', 'a', 'bird', 'in', 'my', 'balcony']
(S
  (NP I)
  (VP
    (VP (V saw) (NP (Det a) (N bird)))
    (PP (P in) (NP (Det my) (N balcony)))))
(S
  (NP I)
  (VP
    (V saw)
    (NP (Det a) (N bird) (PP (P in) (NP (Det my) (N balcony))))))
>>>
```





Practical Number: 8

Aim: Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer, Study WordNetLemmatizer.

PorterStemmer:

Source Code:

```
import nltk
from nltk.stem import PorterStemmer
word_stemmer = PorterStemmer()
print(word_stemmer.stem('writing'))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac8a.py =================
write
>>>
```

LancasterStemmer:

Source Code:

```
import nltk
from nltk.stem import LancasterStemmer
Lanc_stemmer = LancasterStemmer()
print(Lanc_stemmer.stem('writing'))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac8b.py =================
writ
>>>
```

RegexpStemmer:

Source Code:

```
import nltk
from nltk.stem import RegexpStemmer
Reg_stemmer = RegexpStemmer('ing$|s$|e$|able$', min=4)
print(Reg_stemmer.stem('writing'))
```

Output:

```
IDLE Shell 3.10.9
File Edit Shell Debug Options Window Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac8c.py =================
writ
>>>
```

SnowballStemmer:

Source Code:

```
import nltk
from nltk.stem import SnowballStemmer
english_stemmer = SnowballStemmer('english')
print(english_stemmer.stem ('writing'))
```

Output:

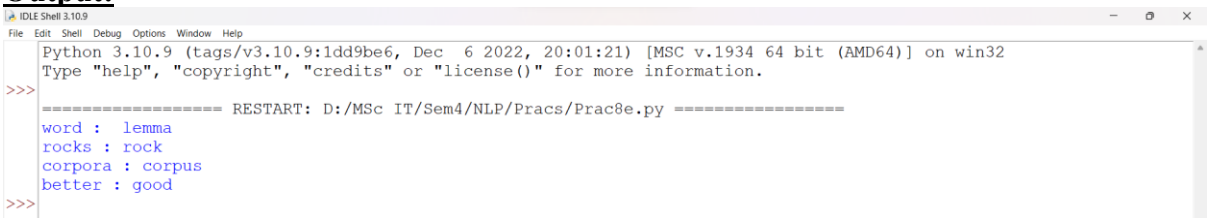
```
IDLE Shell 3.10.9
File Edit Shell Debug Options Window Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac8d.py =================
write
>>>
```

WordNetLemmatizer:

Source Code:

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print("word :\tlemma")
print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))
# a denotes adjective in "pos"
print("better :", lemmatizer.lemmatize("better", pos ="a"))
```

Output:



Practical Number: 9

Aim: Implement Naive Bayes classifier.

Source Code:

```
import pandas as pd
import numpy as np
sms_data = pd.read_csv("spam.csv", encoding='latin-1')
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
stemming = PorterStemmer()
corpus = []
for i in range (0,len(sms_data)):
    s1 = re.sub('[^a-zA-Z]',repl = ' ',string = sms_data['v2'][i])
    s1.lower()
    s1 = s1.split()
    s1 = [stemming.stem(word) for word in s1 if word not in set(stopwords.words('english'))]
    s1 = ' '.join(s1)
    corpus.append(s1)
from sklearn.feature_extraction.text import CountVectorizer
countvectorizer =CountVectorizer()
x = countvectorizer.fit_transform(corpus).toarray()
print(x)
y = sms_data['v1'].values
print(y)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,stratify=y,random_state=2)
#Multinomial Naïve Bayes.
from sklearn.naive_bayes import MultinomialNB
multinomialnb = MultinomialNB()
multinomialnb.fit(x_train,y_train)
```

```
# Predicting on test data:
y_pred = multinomialnb.predict(x_test)
print(y_pred)
#Results of our Models
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
print(classification_report(y_test,y_pred))
print("Accuracy_score: ",accuracy_score(y_test,y_pred))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac9a.py =================
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
['ham' 'ham' 'spam' ... 'ham' 'ham' 'ham']
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
              precision    recall  f1-score   support

         ham       0.99      0.99      0.99      1448
        spam       0.92      0.93      0.92       224

    accuracy                           0.98      1672
   macro avg       0.95      0.96      0.96      1672
weighted avg       0.98      0.98      0.98      1672

Accuracy_score:  0.979066985645933
>>>
```

Practical number: 10

a) Aim: Speech Tagging:

i) Speech tagging using spacy.

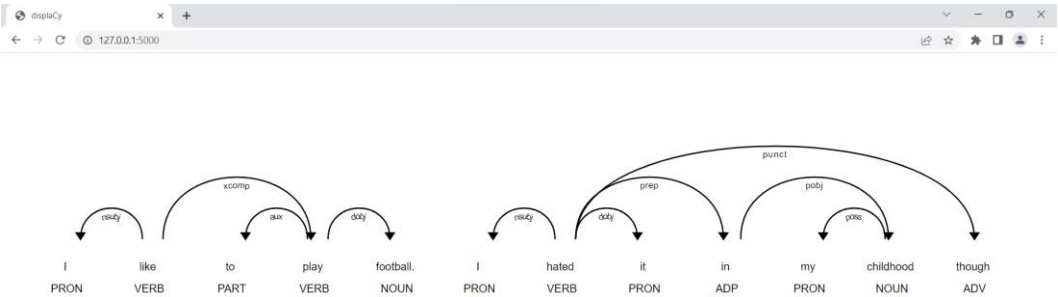
Source Code:

```
import spacy
sp = spacy.load('en_core_web_sm')
sen = sp(u"I like to play football. I hated it in my childhood though")
print(sen.text)
print(sen[7].pos_)
print(sen[7].tag_)
print(spacy.explain(sen[7].tag_))
for word in sen:
    print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}} {spacy.explain(word.tag_)}')
sen = sp(u'Can you google it?')
word = sen[2]
print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}} {spacy.explain(word.tag_)}')
sen = sp(u'Can you search it on google?')
word = sen[5]
print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}} {spacy.explain(word.tag_)}')
#Finding the Number of POS Tags
sen = sp(u"I like to play football. I hated it in my childhood though")
num_pos = sen.count_by(spacy.attrs.POS)
num_pos
for k,v in sorted(num_pos.items()):
    print(f'{k}. {sen.vocab[k].text:{8}}: {v}')
#Visualizing Parts of Speech Tags
from spacy import displacy
sen = sp(u"I like to play football. I hated it in my childhood though")
displacy.serve(sen, style='dep', options={'distance': 120})
```

Output:

```
*IDLE Shell 3.10.9*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac10a.py =================
I like to play football. I hated it in my childhood though
VERB
VBD
verb, past tense
I            PRON       PRP      pronoun, personal
like         VERB       VBP      verb, non-3rd person singular present
to           PART       TO       infinitival "to"
play         VERB       VB       verb, base form
football     NOUN       NN       noun, singular or mass
.            PUNCT      .        punctuation mark, sentence closer
I            PRON       PRP      pronoun, personal
hated        VERB       VBD      verb, past tense
it           PRON       PRP      pronoun, personal
in           ADP        IN       conjunction, subordinating or preposition
my           PRON       PRP$     pronoun, possessive
childhood    NOUN       NN       noun, singular or mass
though       ADV        RB       adverb
google       VERB       VB       verb, base form
google       PROPN      NNP      noun, proper singular
85. ADP     : 1
86. ADV     : 1
92. NOUN    : 2
94. PART    : 1
95. PRON    : 4
97. PUNCT   : 1
100. VERB    : 3

Using the 'dep' visualizer
Serving on http://0.0.0.0:5000 ...

127.0.0.1 - - [12/Apr/2023 19:03:47] "GET / HTTP/1.1" 200 9471
127.0.0.1 - - [12/Apr/2023 19:03:48] "GET /favicon.ico HTTP/1.1" 200 9471
```

To view the dependency tree, type the following address in your browser: http://127.0.0.1:5000/. You will see the following dependency tree:



ii) Speech tagging using nltk

Source Code:

```
import nltk
from nltk.corpus import state_union
from nltk.tokenize import PunktSentenceTokenizer
#create our training and testing data:
train_text = state_union.raw("2005-GWBush.txt")
sample_text = state_union.raw("2006-GWBush.txt")
#train the Punkt tokenizer like:
custom_sent_tokenizer = PunktSentenceTokenizer(train_text)
# tokenize:
tokenized = custom_sent_tokenizer.tokenize(sample_text)
def process_content():
    try:
        for i in tokenized[:2]:
            words = nltk.word_tokenize(i)
            tagged = nltk.pos_tag(words)
        print(tagged)
    except Exception as e:
        print(str(e))
process_content()
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: D:\MSc IT\Sem4\NLP\Pracs\Prac10aii.py ================
[('Mr.', 'NNP'), ('Speaker', 'NNP'), (',', ','), ('Vice', 'NNP'), ('President', 'NNP'), ('Cheney', 'NNP'), (',', '
,'), ('members', 'NNS'), ('of', 'IN'), ('Congress', 'NNP'), (',', ','), ('members', 'NNS'), ('of', 'IN'), ('the',
'DT'), ('Supreme', 'NNP'), ('Court', 'NNP'), ('and', 'CC'), ('diplomatic', 'JJ'), ('corps', 'NN'), (',', ','), ('d
istinguished', 'JJ'), ('guests', 'NNS'), (',', ','), ('and', 'CC'), ('fellow', 'JJ'), ('citizens', 'NNS'), (':', '
:'), ('Today', 'VB'), ('our', 'PRP$'), ('nation', 'NN'), ('lost', 'VBD'), ('a', 'DT'), ('beloved', 'VBN'), (',', '
,'), ('graceful', 'JJ'), (',', ','), ('courageous', 'JJ'), ('woman', 'NN'), ('who', 'WP'), ('called', 'VBD'), ('Am
erica', 'NNP'), ('to', 'TO'), ('its', 'PRP$'), ('founding', 'NN'), ('ideals', 'NNS'), ('and', 'CC'), ('carried', '
VBD'), ('on', 'IN'), ('a', 'DT'), ('noble', 'JJ'), ('dream', 'NN'), ('.', '.')]
>>>
```

b) Aim: Statistical parsing:

i) Usage of Give and Gave in the Penn Treebank sample.

Source Code:

```
#Usage of Give and Gave in the Penn Treebank sample
import nltk
import nltk.parse.viterbi
import nltk.parse.pchart
def give(t):
    return t.label() == 'VP' and len(t) > 2 and t[1].label() == 'NP'\
        and (t[2].label() == 'PP-DTV' or t[2].label() == 'NP')\
        and ('give' in t[0].leaves() or 'gave' in t[0].leaves())
def sent(t):
    return ' '.join(token for token in t.leaves() if token[0] not in '*-0')
def print_node(t, width):
    output = "%s %s: %s / %s: %s" %\
            (sent(t[0]), t[1].label(), sent(t[1]), t[2].label(), sent(t[2]))
    if len(output) > width:
        output = output[:width] + "..."
    print(output)
for tree in nltk.corpus.treebank.parsed_sents():
    for t in tree.subtrees(give):
        print_node(t, 72)
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac10b.py ====
gave NP: the chefs / NP: a standing ovation
give NP: advertisers / NP: discounts for maintaining or increasing ad sp...
give NP: it / PP-DTV: to the politicians
gave NP: them / NP: similar help
give NP: them / NP:
give NP: only French history questions / PP-DTV: to students in a Europe...
give NP: federal judges / NP: a raise
give NP: consumers / NP: the straight scoop on the U.S. waste crisis
gave NP: Mitsui / NP: access to a high-tech medical product
give NP: Mitsubishi / NP: a window on the U.S. glass industry
give NP: much thought / PP-DTV: to the rates she was receiving , nor to ...
give NP: your Foster Savings Institution / NP: the gift of hope and free...
give NP: market operators / NP: the authority to suspend trading in futu...
gave NP: quick approval / PP-DTV: to $ 3.18 billion in supplemental appr...
give NP: the Transportation Department / NP: up to 50 days to review any...
give NP: the president / NP: such power
give NP: me / NP: the heebie-jeebies
give NP: holders / NP: the right , but not the obligation , to buy a cal...
gave NP: Mr. Thomas / NP: only a `` qualified '' rating , rather than ``...
give NP: the president / NP: line-item veto power
>>>
```

ii) Probabilistic parser

Source code:

```
import nltk
from nltk import PCFG
grammar = PCFG.fromstring('''
NP -> NNS [0.5] | JJ NNS [0.3] | NP CC NP [0.2]
NNS -> "men" [0.1] | "women" [0.2] | "children" [0.3] | NNS CC NNS [0.4]
JJ -> "old" [0.4] | "young" [0.6]
CC -> "and" [0.9] | "or" [0.1]
''')
print(grammar)
viterbi_parser = nltk.ViterbiParser(grammar)
token = "old men and women".split()
obj = viterbi_parser.parse(token)
print("Output: ")
for x in obj:
    print(x)
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac10bii.py ================
Grammar with 11 productions (start state = NP)
    NP -> NNS [0.5]
    NP -> JJ NNS [0.3]
    NP -> NP CC NP [0.2]
    NNS -> 'men' [0.1]
    NNS -> 'women' [0.2]
    NNS -> 'children' [0.3]
    NNS -> NNS CC NNS [0.4]
    JJ -> 'old' [0.4]
    JJ -> 'young' [0.6]
    CC -> 'and' [0.9]
    CC -> 'or' [0.1]
Output:
(NP (JJ old) (NNS (NNS men) (CC and) (NNS women))) (p=0.000864)
>>>
```

c. Aim: Malt parsing:

Parse a sentence and draw a tree using malt parsing.

Note: 1) Java should be installed.

2) maltparser-1.7.2 zip file should be copied in
C:\Users\hp\AppData\Local\Programs\Python\Python311 folder and should be

Extracted in the same folder.

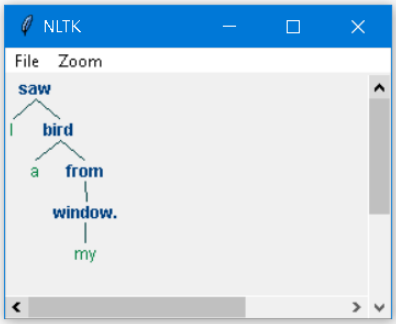
3) engmalt.linear-1.7.mco file should be copied to
C:\Users\hp\AppData\Local\Programs\Python\Python311 folder

Source code:

```
# copy maltparser-1.7.2(unzipped version) and engmalt.linear-1.7.mco files to

C:\Users\hp\AppData\Local\Programs\Python\Python311 folder

# java should be installed

# environment variables should be set – MALT_PARSER –
C:\Users\hp\AppData\Local\Programs\Python\Python311\maltparser-1.7.2 and

MALT_MODEL – C:\Users\hp\AppData\Local\Programs\Python\Python311\engmalt.linear-
1.7.mco

From nltk.parse import malt

Mp = malt.MaltParser('maltparser-1.7.2', 'engmalt.linear-1.7.mco')#file

T = mp.parse_one('I saw a bird from my window.'.split()).tree()

Print(t)

t.draw()
```

Output:

```
*Python 3.9.0b3 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.9.0b3 (tags/v3.9.0b3:b484871, Jun  9 2020, 20:36:59) [MSC v.1924 64 bit
 (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\hp\AppData\Local\Programs\Python\Python311\Lib\site-packages
\nltk\parse\pract10c.py
(saw I (bird a (from (window. my))))
```



Practical number: 11

a) Aim: Multiword Expressions in NLP.

Source code:

```
from nltk.tokenize import MWETokenizer
from nltk import sent_tokenize, word_tokenize
s = '''Good cake cost Rs.1500\kg in Mumbai. Please buy me one of them.\n\nThanks.'''
mwe = MWETokenizer([('New', 'York'), ('Hong', 'Kong')], separator='_')
for sent in sent_tokenize(s):
    print(mwe.tokenize(word_tokenize(sent)))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac11a.py ====
['Good', 'cake', 'cost', 'Rs.1500\\kg', 'in', 'Mumbai', '.']
['Please', 'buy', 'me', 'one', 'of', 'them', '.']
['Thanks', '.']
>>>
```

b) Aim: Normalized Web Distance and Word Similarity.

Source code:

```
import numpy as np
import re
import warnings
warnings.filterwarnings("ignore")
import textdistance # pip install textdistance
# we will need scikit-learn>=0.21
import sklearn #pip install sklearn
from sklearn.cluster import AgglomerativeClustering
texts = [
'Reliance supermarket', 'Reliance hypermarket', 'Reliance', 'Reliance', 'Reliance downtown',
'Relianc market',
'Mumbai', 'Mumbai Hyper', 'Mumbai dxb', 'mumbai airport',
'k.m trading', 'KM Trading', 'KM trade', 'K.M. Trading', 'KM.Trading'
]
def normalize(text):
    """ Keep only lower-cased text and numbers"""
    return re.sub('[^a-z0-9]+', ' ', text.lower())
def group_texts(texts, threshold=0.4):
    """ Replace each text with the representative of its cluster"""
    normalized_texts = np.array([normalize(text) for text in texts])
    distances = 1 - np.array([
        [textdistance.jaro_winkler(one, another) for one in normalized_texts]
        for another in normalized_texts
])
    clustering = AgglomerativeClustering(
        distance_threshold=threshold, # this parameter needs to be tuned carefully
        affinity="precomputed", linkage="complete", n_clusters=None
    ).fit(distances)
```

```
    centers = dict()
    for cluster_id in set(clustering.labels_):
        index = clustering.labels_ == cluster_id
        centrality = distances[:, index][index].sum(axis=1)
        centers[cluster_id] = normalized_texts[index][centrality.argmin()]
    return [centers[i] for i in clustering.labels_]
print(group_texts(texts))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac11b.py =================
['reliance', 'reliance', 'reliance', 'reliance', 'reliance', 'reliance', 'mumbai', 'mumbai', 'mumbai', 'mumbai', '
km trading', 'km trading', 'km trading', 'km trading', 'km trading']
>>>
```

c) Aim: Word Sense Disambiguation.

Source code:

```
from nltk.corpus import wordnet as wn
def get_first_sense(word, pos=None):
    if pos:
        synsets = wn.synsets(word,pos)
    else:
        synsets = wn.synsets(word)
    return synsets[0]
best_synset = get_first_sense('bank')
print ('%s: %s' % (best_synset.name, best_synset.definition))
best_synset = get_first_sense('set','n')
print ('%s: %s' % (best_synset.name, best_synset.definition))
best_synset = get_first_sense('set','v')
print ('%s: %s' % (best_synset.name, best_synset.definition))
```

Output:

```
IDLE Shell 3.10.9
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: D:/MSc IT/Sem4/NLP/Pracs/Prac11c.py =================
<bound method Synset.name of Synset('bank.n.01')>: <bound method Synset.definiti
on of Synset('bank.n.01')>
<bound method Synset.name of Synset('set.n.01')>: <bound method Synset.definitio
n of Synset('set.n.01')>
<bound method Synset.name of Synset('put.v.01')>: <bound method Synset.definitio
n of Synset('put.v.01')>
>>>
```